UCRL-WEB-200719

EZSTORAGE

Table of Contents

Preface

Scope: EZSTORAGE explains how to transfer files between machines where you work (mostly LC production machines) and LLNL's High Performance Storage System (or STORAGE, LC's central file-storage archive). Transferring files using FTP and NFT as well as connecting to STORAGE from various locations is discussed. EZSTORAGE also tells how to overcome the most common problems encountered when storing and retrieving your files, including sharing stored files and storing Macintosh files. Three customized (LC-only) storage-assistance tools to manage and monitor the groups and permissions of your stored files are also introduced (LSTORAGE, CHMODSTG, and CHGRPSTG), along with a fourth local tool (HTAR) that supports the fast, efficient storage of very large archive (TAR-like library) files.

Availability: When the programs described here are limited by machine, those limits are included in their explanation. Otherwise, they run under any LC UNIX system.

Consultant: For help contact the LC customer service and support hotline at 925-422-4531 (open e-mail: lc-hotline@llnl.gov, SCF e-mail: lc-hotline@pop.llnl.gov).

Printing: The print file for this document can be found at

```
OCF: http://www.llnl.gov/LCdocs/ezstorage/ezstorage.pdf
SCF: https://lc.llnl.gov/LCdocs/ezstorage/ezstorage_scf.pdf
```

Introduction

This manual provides a basic guide to effectively storing and archiving files from LC computers by using the High Performance Storage System (HPSS). Its goal is to introduce relevant background information, describe storage interface options, and give an overview of the basic commands used for storage. The HPSS Manual (URL: http://www.llnl.gov/LCdocs/hpss) provides a detailed discussion of the STORAGE system and its specialized features.

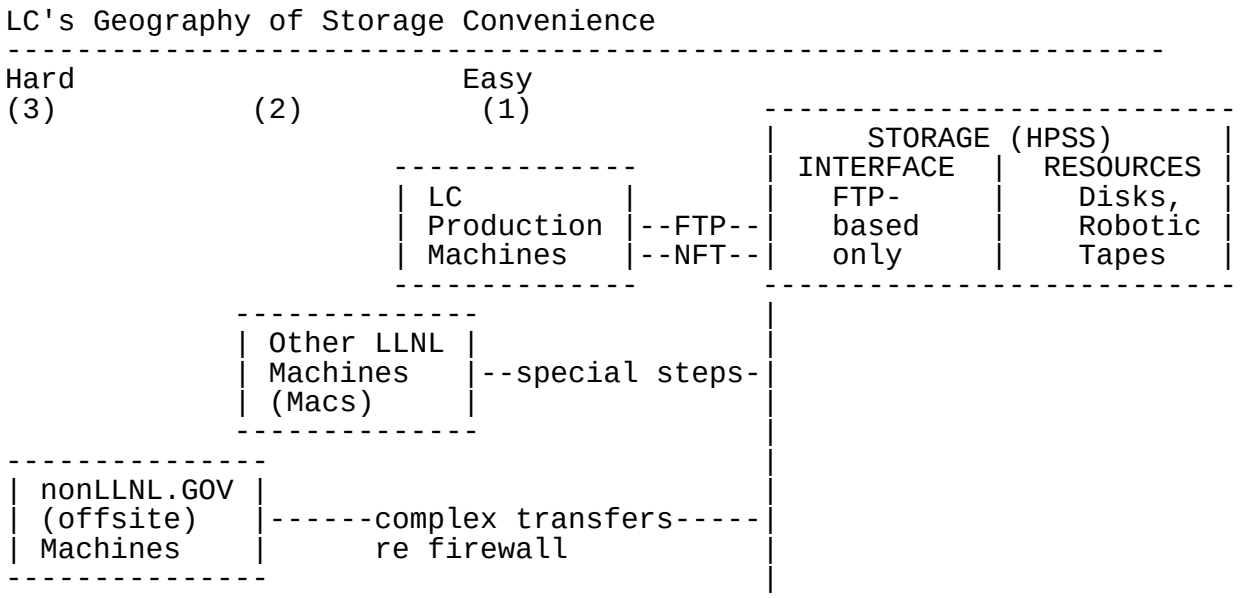
EZSTORAGE first provides an overview (page 5) of the strengths and weaknesses of LC's storage system, including the software interfaces (page 8) compatible with STORAGE, a summary of STORAGE commands (page 7), and accessing STORAGE (page 10) from various locations. The second section explains how to save files using File Transfer Protocol (page 12)(FTP). The third section explains how to save files using Network File Transfer (page 18) (NFT). The next section answers the common but complex question of how to share stored files. (page 21) A subsequent section details file-storage problems (page 25) that may occur when moving files to and from a Macintosh computer. Finally, a concluding section introduces three special storage-assistance (page 29) software tools to simplify managing your stored files, along with another local tool (HTAR) that efficiently stores large archive (library) files or retrieves members from within them while stored.

Reference manuals are available to provide detailed technical instructions on the tools and techniques introduced in EZSTORAGE, including manuals on FTP (URL: http://www.llnl.gov/LCdocs/ftp), NFT (URL: http://www.llnl.gov/LCdocs/nft), HPSS (URL: http://www.llnl.gov/LCdocs/hpss), HTAR (URL: http://www.llnl.gov/LCdocs/htar), and the local Firewall (URL: http://www.llnl.gov/LCdocs/firewall). Additionally the document EZFILES (URL: http://www.llnl.gov/LCdocs/ezfiles) is a basic guide for using local directories and for general file-handling software at LC. (Because "secure FTP," called SFTP, relies on a different server than standard FTP, you can *not* use SFTP to store files or retrieve stored files at LLNL.)

If FTP transfer rates and reliability are important concerns for you (because you store large files, for example), then you can monitor recent FTP performance between many pairs of network nodes (including storage.llnl.gov) by using LC's NETMON web site (see the NETMON Reference Manual (URL: http://www.llnl.gov/LCdocs/netmon) for details). NETMON automatically reports all FTP traffic divided by file size, using just the same file-size categories that HPSS uses to define each storage "class of service" at LC (see "Copies in Storage" (page 11) below), and you can choose Mbyte or Mbit as the reporting unit.

Overview

Reliable, massive, archival data storage is a crucial part of any effective high-performance computing environment. The good news is that LC has such storage (officially called the High Performance Storage System, HPSS). The bad news is that layers of historically complex, asymmetric, security-driven interface features can make using storage easy in some circumstances but almost impossible in others. This diagram shows storage in its user context at LC:



Note first (right-hand side) that although the actual disk and tape resources for storing files at LC are large and elaborate, the user interface here is constrained to use the FTP daemons exclusively. This means that all your interactions with storage occur through FTP (or local FTP extensions such as NFT, parallel FTP (PFTP), or HTAR). FTP is standardized and easy to learn, but for many typical storage tasks it is inept or profoundly inelegant (the file-sharing (page 21) section below provides an extended example). Since SFTP talks to a different, nonFTP daemon, you cannot store or retrieve files using SFTP.

Storing files from (or retrieving stored files to) LC production machines, open or secure, is a mainstream storage mission (see (1) in the diagram), easy to perform and very reliable. Using file storage in this context avoids quotas on user home directories, avoids purges of files on temporary work disks, and provides virtually unlimited capacity for managing data or computational output. Transfer rates are fast (up to gigabytes/sec) and FTP connections are very reliable. Customized FTP interfaces to handle special storage needs (such as NFT for persistent storage transfers or HTAR for efficiently making large archives directly in storage) are available here too.

Storing files from (or retreiving stored files to) other LLNL machines, in particular Macintosh desktop or portable computers (see (2) in the diagram), is more complex. Features of special FTP clients (such as Fetch) together with the need to protect unusual file formats during transfer to or from storage call for taking extra steps, described in a separate section (page 25) below.

Finally, storing files from (or retrieving stored files to) nonLLNL.GOV machines (see (3) in the diagram), such as computers at other sites or the workstations of distant ASCI collaborators, is the most complicated of the three situations. It requires either using a two-stage process or running extra enabling software (such as VPN). This may involve running FTP twice, or using nonFTP transfers to an LC production machine before actually storing the files with FTP (run on an LC machine). The details, suggested in a later section (page 10), call for entire separate manuals to explain thoroughly, and nevertheless often change to reflect unsettled security goals.

So read the sections of EZSTORAGE with this guiding "geography of storage convenience" in mind. Some storage tasks from some vantage points are easy and the introductory material here easily suffices to cover them. Other storage tasks from other vantage points are elaborate or barrier-prone, and this guide can only suggest the basic approach that might require much elaboration before you finally achieve success.

Storage Summary

This section briefly summarizes the chief storage-system constraints and tells how to perform the most important file-storage tasks at LC. We suggest you save it for ready reference.

Storage System Constraints:

```
Largest allowed file size: 512 Gbyte (using FTP/NFT interface)
                           8 Gbyte/member       |(using HTAR
                           no limit/archive    | interface)
Longest file name: 255 characters
Problem characters in file names:
 Treated as file filters: ? * {a,b}
 Forbidden FIRST characters: - ! ~
 Forbidden in any position: ' " : ; { } , /
```

Commands for Common File-Storage Tasks:

TASK:	**FTP**	**NFT**
Connect to storage:	ftp storage	nft
Make storage directory:	mkdir *dr*	(same)
Change storage directories:	cd *dr*	(same)
Store a file:	put *fl*	(same)
Retrieve a stored file:	get *fl*	(same)
Retrieve from within a stored archive:	See HTAR	See HTAR
Delete a stored file:	delete *fl*	(same)
List stored files:	dir	(same)
Change permissions (to *nnn*):	quote site chmod *nnn fl*	chmod *nnn fl*
Change "class of service" (COS):	site setcos *nnn*	(none)
Change access control list (ACL):	(none)	acladd etc.
Start migration of stored file from tape:	quote site stage *fl*	(none)
Control file overwriting:		
.....Prevent overwriting	(none)	noclobber
.....Allow overwriting	(default)	clobber

STORAGE Interfaces

FTP: General File Transfers

FTP is the most well known and generally supported file-transfer utility, therefore FTP clients and (server) daemons are available on all LC production and special-purpose machines in both the open and secure environments. FTP is the standard interface to the LC archival file storage system (both open and secure). When you run FTP (on an OTP or DCE-passworded LC machine) with STORAGE as the target host, access is "preauthenticated" and you are NOT prompted for your password. Also, on all LC production machines (but not necessarily on other LC machines), a parallel FTP client (equivalent to PFTP) is now the default. All files that are 1 Mbyte or larger automatically move to or from storage using parallel FTP.

A concise summary of how to use FTP commands and features to store files, with annotated examples, is found in the Using FTP (page 12) section of this document (along with a subsection on known pitfalls). For a detailed discussion of the user commands, software responses, and error codes for the FTP file-transfer utility consult the FTP Reference Manual. (URL: http://www.llnl.gov/LCdocs/ftp)

NFT: Locally Enhanced Transfers

NFT is available on all LC production machines (open and secure, but not on some special-purpose hosts) and is a locally developed file transfer tool. Although NFT uses standard FTP daemons to carry out its file transfers, it offers such enhanced features as:

- A special NFT server preauthenticates all NFT transfers, so all NFT executions are passwordless.
- NFT elaborately tracks and numbers all transfers. It automatically persists if system problems delay storing any file, and it keeps detailed records of your file-storage successes and problems.
- Input from and output to files is easy, and NFT's command syntax (unlike FTP's) lends itself to practical use in scripts and batch jobs.
- Some NFT commands especially facilitate transfers to and from STORAGE (some users regard NFT as primarily a file-storage rather than a general file-transfer tool). NFT but *not* FTP offers five commands that manipulate access control lists (ACLs) on stored files.

A concise, task-oriented summary of how to use NFT commands and features, with annotated typical examples is found in the Using NFT (page 18) section of this document. For a complete analysis of NFT syntax and special features, along with a thorough alphabetical command dictionary, consult the NFT Reference Manual (URL: http://www.llnl.gov/LCdocs/nft).

Additional Interfaces

In addition to FTP and NFT, there are also other local tools that use FTP daemons for file transfer to STORAGE through graphical interfaces. One for workstations is XDIR; one for Macintosh computers is Fetch. Another, called HTAR (page 38), has been designed at LC primarily to efficiently transfer very large archive (TAR-like library) files to and from STORAGE on LC production machines, or to extract member files from within still-stored archives (you can optionally use HTAR for similar *non*STORAGE transfers too).

NOTE: At LC, currently only STORAGE interfaces based on FTP daemons are supported (such as FTP, PFTP, NFT, HTAR, XDIR, and Fetch). The Network File System (NFS) and IBM SP Parallel I/O File System (PIOFS) interfaces are NOT available. And despite its name, "secure FTP" (SFTP) relies on the SSHD2 daemon, not the standard FTP daemon, so you can NOT store or retrieve stored files at LLNL using SFTP as an interface. However, executing FTP on any LC production machine (but not necessarily on any other LC machines) is equivalent to executing PFTP (Parallel FTP) by default. Executing the PFTP client overtly gives you access to nine special parallel-transfer commands (such as PPUT and PGET). These extra PFTP commands are unnecessary on LC production machines (where ordinary FTP performs parallel transfers to and from storage automatically) but if you use STORAGE at *other* ASCI (tri-lab) sites you may need to invoke them (see the HPSS User Guide (URL: http://www.llnl.gov/LCdocs/hpss) for details).

Accessing STORAGE

Accessing STORAGE is most easily done from an LC production machine. Offsite users will encounter difficulties in connecting to STORAGE because of interface limitations as well as intentional security barriers to easy use.

When onsite, NFT and FTP can be used to transfer files to and from STORAGE. The NFT interface is supported only by LC machines and can only be used to transfer files between them. It is not possible to use NFT from a machine outside 134.n.n.n (including other llnl.gov machines and all onsite desktop machines).

Offsite users can only use FTP. However, LC's firewall totally blocks all FTP traffic from every host outside the llnl.gov domain. To transfer files from machines outside llnl.gov to any LC machine, outside-the-firewall users have three choices:

(1) Log on to an LC production machine, then execute FTP on that machine and connect back to the outside machine where the sought files reside, using GET to retrieve them. This approach poses known problems for Macintosh files, and suggested solutions appear in the Macintosh problems (page 25) section. It also requires an FTP server (not just a client) running on the outside machine, a problem for some workstations. As a second stage, you must then run FTP on the llnl.gov machine again to transfer the files to storage.

(2) Run secure copy SCP (described in EZOUTPUT (URL: http://www.llnl.gov/LCdocs/ezoutput)) instead of FTP to transfer files toward an open-network LC machine. You must have previously installed SSH on the outside machine (LC's firewall allows SSH and SCP traffic from any IPA-authenticated outside host). Since storage.llnl.gov has no SCP server, however, you must again run FTP on the LC machine as a second stage to actually transfer your files to storage.

(3) Before you run FTP on your outside-the-firewall machine, get, install, configure, and execute a Virtual Private Network (VPN) client on that machine. Contact the LC Hotline to see if you are authorized to run a VPN client for access to LLNL. A VPN client borrows an llnl.gov IP address for your machine while it runs, and LC has confirmed that if you run VPN and FTP together under Windows98, you can directly transfer files to storage.llnl.gov from outside the firewall (no staging to an LC production machine is needed). But you may encounter vendor-compatibility problems with other versions of Windows or with other operating systems. See LC's Firewall and SSH Guide (URL: http://www.llnl.gov/LCdocs/firewall) for full instructions on the fairly complex process of getting and using VPN to enable FTP. (You cannot use SFTP through VPN to store or retrieve stored files.)

[A former alternative choice, obtaining a CRYPTOCard (URL: http://www.llnl.gov/LCdocs/firewall) for authentification and contacting the firewall gateway (gw-lc.llnl.gov) before opening an (indirect) FTP connection to any other LC machine, was discontinued for public use in April, 2000.]

Additionally, LC's firewall now blocks all TELNET (interactive login) traffic originating from nonLC machines (from all machines outside 134.n.n.n, even from other llnl.gov machines), although it still allows IPA-authenticated SSH traffic originating from those machines. Detailed instructions for the three choices mentioned above, as well as a concise but thorough SSH overview, including role, annotated setup steps, basic execute lines, and troubleshooting tips, are available in LC's Firewall and SSH Guide (URL:

http://www.llnl.gov/LCdocs/firewall). This manual is posted on both the open and secure LC documentation web servers.

Copies in Storage

Some files may be so important to your project that you want to store separate, duplicate copies on independent storage media (at LC, this means separate tape cartridges). LC's OCF and SCF storage systems offer such dual-copy storage using the "class of service" (COS) concept.

The storage server(s) assign to every incoming file a class of service (COS) based on the file's size and the client that writes it:

- Files written with FTP or NFT that are smaller than 32 Mbyte are *automatically* assigned a COS that provides two separate copies on separate storage tapes. For these files you never need to request duplicate storage.
- Files written with FTP or NFT that are 32 Mbyte or larger are assigned a COS that stores only a single copy. For mission critical files in this category you can request dual-copy storage by using the FTP command

  ```
  site setcos 150
  ```

  before you PUT the large file(s) into HPSS. (There is no similar NFT command to change the default COS.)
- Files written with HTAR, regardless of their size, always get a default COS that stores only a single copy. For mission critical files written with HTAR you can request dual-copy storage by using the (uppercase) command

  ```
  -Y 150
  ```

  on the HTAR execute line that creates your stored archive (this overrides the HTAR_COS environment variable).

(For more COS technical details, consult the SETCOS section of LC's HPSS User Guide. (URL: http://www.llnl.gov/LCdocs/hpss) FTP monitoring with NETMON uses the same COS file-size distinctions; see the next section.)

Using FTP

Basic FTP Commands

FTP is a widely used file-transfer utility because it supports transfers between any machines that recognize the TCP/IP protocols, even if they have different architectures or operating systems. You must, however, log in to the remote machine and transfer the files interactively, using your (remote) password.

Because of the need for fast, reliable file transfers to and from STORAGE (i.e., storage.llnl.gov), that host uses special FTP servers and other LC machines use special FTP clients that can preauthorize your FTP login to STORAGE (only), so that no password is requested. All LC production machines (IBM SPs and Compaqs) offer passwordless (preauthenticated) FTP service to STORAGE, and so also do a few specialty machines such as LUCY. The usage is the same as standard FTP, except for omiting the password request. Note also that on production LC machines, NFT, which favors the STORAGE system in several ways, also offers passwordless file transfer to and from STORAGE (see the NFT (page 18) section for details). Furthermore, on all LC production machines (but not necessarily on other LC machines), a parallel FTP client (equivalent to PFTP) is now the default. See the FTP Reference Manual (URL: http://www.llnl.gov/LCdocs/ftp) for instructions on invoking a nondefault nonparallel FTP client, which is less verbose.

If FTP transfer rates and reliability are important concerns for you (because you store large files, for example), then you can monitor recent FTP performance between many pairs of network nodes (including storage.llnl.gov) by using LC's NETMON web site (see the NETMON Reference Manual (URL: http://www.llnl.gov/LCdocs/netmon) for details). NETMON automatically reports all FTP traffic divided by file size, using just the same file-size categories that HPSS uses to define each storage "class of service" at LC (see the "Copies in Storage" (page 11) section above), and you can choose Mbyte or Mbit as the reporting unit.

Most FTP implementations support many commands, but not always the same ones. The standard FTP commands, with their syntax and error codes, are detailed in the FTP Reference Manual (URL: http://www.llnl.gov/LCdocs/ftp). The following FTP commands are the most commonly used ones for basic file transfer:

cd *pathname* — changes directories (on the remote machine) to the one specified by *pathname*. By default, FTP GETs files from and PUTs files to the home directory of the remote machine, so you must change directories with CD if you need to transfer them to or from somewhere else.

pwd — reports the current working directory's pathname on the remote machine (to confirm uses of CD).

dir — lists the names and attributes of files in the current working directory on the remote machine.

get *remotefile* [*localfile*]

retrieves *remotefile* and places it in the current directory of the local machine (where you are running FTP). The incoming file is called *remotefile* by default, or called

localfile if you specify a name. (Use HTAR (page 38) instead if you want to retrieve a member file from within a still-stored archive.)

mget *filelist* generalizes the GET command to transfer all the files in *filelist*, a blank-delimited list of remote files to retrieve to the current directory (where you are running FTP). MGET accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding transfer.

parallel [LLNL only] enables parallel file transfers on LC production machines. But remember that parallel file transfers are already ON by default to or from STORAGE for all files over 1 Mbyte (so typing PARALLEL here just reports the stripe width and block size).

put *localfile* [*remotefile*]

copies *localfile* into the current working directory of the remote machine you have logged in to with FTP. The outwardly transferred file is called *localfile* by default, or called *remotefile* if you specify a name.

mput *filelist* generalizes the PUT command to transfer all the files in *filelist*, a blank-delimited list of local files to copy to the home directory of the remote machine that you logged in to with FTP. MPUT accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding transfer.

delete *remotefile*

removes *remotefile* from the current working directory of the remote machine you have logged in to with FTP. Use DIR to confirm your deletion.

mdelete *filelist* generalizes the DELETE command to remove all the files in *filelist*, a blank-delimited list of remote files to delete from the current working directory on the remote machine. MDELETE accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding deletion. WARNING: see the known pitfall of using MDELETE with wildcards to delete files from the LC storage system (subsection below (page 16)).

help [*command*]

lists the commands supported by the implementation of FTP that you are running, or (with an argument) briefly describes one command.

quit closes your remote session and terminates FTP.

FTP Example

This annotated example shows a typical file transfer to and from STORAGE using FTP.

GOAL: To transfer files to and from STORAGE interactively using FTP (the default parallel FTP client). In this case, the local machine on which the user (JANE) executes the FTP client is GPS17, and the remote machine that files are saved to and retrieved from is STORAGE

STRATEGY: (1) The user runs FTP (on GPS17) with storage.llnl.gov as the remote machine's domain name.
(2) Because STORAGE is a special destination, its FTP server preauthenticates JANE and asks for neither her user name nor her password (most other destinations ask for both). This dialog is more verbose than that for most FTP sites and automatically enables parallel file transfers.
(3) At the ftp> prompt, the user GETs file TEST5 (copies it from STORAGE to GPS17). Note that FTP uses four simultaneous parallel stripes to move the file, automatically.
(4) At the next ftp> prompt, the user PUTs file TABLE.DAT (copies it from GPS17 to STORAGE). Note that again FTP automatically uses four parallel stripes to move the file more quickly.
(5) When the file transfers are done and confirmed, the user QUITs FTP.

```
(1) ftp storage.llnl.gov
    Connected to toofast15.llnl.gov
    220-NOTICE TO USERS [long legal disclaimer here...]
    220 toofast15 FTP server (HPSS 4.1 PFTPD V1.1.45
        Tue Sep 5  14:06:03 PDT 2000) ready.
    334 Using authentication type GSSAPI; ADAT must follow
    GSSAPI accepted as authentication type
    GSSAPI authentication succeeded

(2) Preauthenticated FTP to toofast15.llnl.gov as jane:
    230 User /.../spectrum.llnl.gov/jane logged in.
    Remote system type is UNIX.
    Using binary mode to transfer files.
    Daemon supports Parallel Features.
    -Auto-Parallel Substitution Enabled
    Parallel stripe width set to (4).
    Parallel block size set to (1048576).
    Multinode is Disabled.

(3) ftp> get test5
    200 Command complete (1827811,test5,0,1,131072).
    200 Command complete.
    200 Command complete.
    200 Command complete.
    200 Command complete.
    150 Transfer starting.
    226 Transfer complete. (moved = 1827811).
    1827811 bytes received in 0.20 seconds (9.10 Mbytes/s)
    200 Command complete.
```

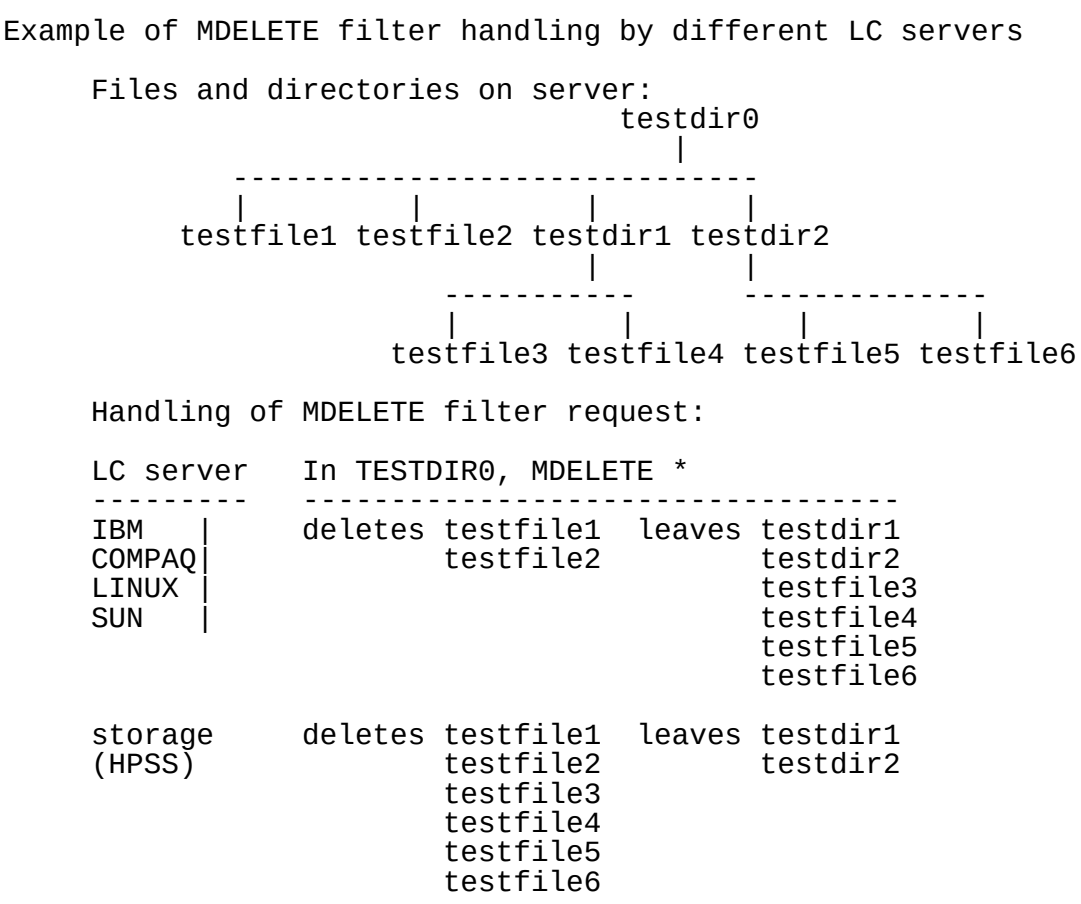
```
(4) ftp> put table.dat
    200 Command complete.(7311244,table.dat,0,1,4194304).
    200 Command complete.
    200 Command complete.
    200 Command complete.
    200 Command complete.
    150 Transfer starting.
    200 PORT command successful.
    226 Transfer complete. (moved = 7311244).
    7311244 bytes sent in 1.85 seconds (3.96 Mbytes/s)
    200 Command complete.

(5) ftp> quit
    221 Goodbye.
```

FTP Pitfalls (with Storage)

FTP's "M" commands (such as MGET, MPUT, and MDELETE) process multiple files by using as their argument either an explicit file list or a file filter (an implicit file list specified with one or more UNIX wildcards or metacharacters, such as ? or *). The UNIX shell where your FTP *client* runs expands each file filter (or "ambiguous file reference," such as TEST*) into a list of names for FTP to process.

But how that list is processed depends on the FTP *server* that receives it. Different servers may process the expanded filter differently, with different file-transfer results. Thus the same FTP "M" command may behave differently on different servers. When your valuable stored files are involved, and when the FTP command is MDELETE (or any file-removal command), the results can sometimes be very inappropriate.

The diagram below illustrates how problems can arise because the LC storage server treats MDELETE filters differently than do other FTP servers at LC.



Only the server on the LC storage system interprets MDELETE filters recursively, and removes all (matching) files not only in the current working directory but also in the directory children of that directory as well. (Interestingly, the LC storage server treats only MDELETE so aggressively; MPUT and MGET still act just within the current directory.) You should always pay attention to this known pitfall when deleting multiple stored files at LC using FTP as your storage interface.

You can work around this broad interpretation of FTP's MDELETE command by the LC storage server in several ways:

- Use as your MDELETE argument an explicit list of files instead of a file filter with metacharacters.
- Use a more restrictive filter (certainly more restrictive than * alone), carefully crafted to select only the files in the current working directory in storage (if your file-naming scheme allows).
- Use NFT (page 18) instead of FTP as your storage interface. NFT has no MDELETE command, and its separate storage server interprets DELETE * to remove only files in the current working directory (not in any child directories). You must invoke the -R suboption explicitly to make the NFT commands DELETE and RMDIR behave recursively.

Using NFT

NFT Command Syntax

NFT (unlike FTP) was designed for the LC environment and it has two special features that affect how you can use it.

First, all NFT file transfers involve not only the donor and the receiver machines you specify (overtly or by default), but also a third invisible machine running locally developed software (ENDEAVOR) dedicated to failure detection and recovery and NFT job tracking. If a problem prevents a file from being sent or received immediately, ENDEAVOR automatically remembers the request and persists in trying to complete it later, recording its results for you to verify if needed.

Second, NFT assumes that the remote host in all file transfers is the LC storage system (storage.llnl.gov) unless you specify otherwise. The existence of such a default remote host means that:
(1) There are two types of NFT commands, (one for general file transfers among any hosts that NFT serves, and one for those that do not accept NFT's usual host-specifying syntax because they default to storage transfers) and,
(2) The syntax of NFT commands assumes local-to-storage transfers unless you specify otherwise.

NFT also has many special features that suit it for use in batch jobs and scripts (unlike FTP). The NFT Reference Manual (URL: http://www.llnl.gov/LCdocs/nft) explains those unusual features, while its basic commands pertaining to STORAGE are summarized here. Unlike FTP, NFT transfers involve no long legal preamble and no increased verbosity when connecting to a parallel FTP server such as STORAGE. Also unlike FTP, NFT provides five commands dedicated to manipulating the (optional) access control lists (ACLs) on stored files (this is the only way that those who are not HPSS system administrators can change ACLs in the storage system). (If you want to retrieve a member file from within a still-stored archive, use HTAR (page 38) instead.)

NFT Commands by Task

This chart shows the interactive NFT commands that perform the most common file-transfer tasks. For a detailed list of every NFT command, see the Command Dictionary (URL: http://www.llnl.gov/LCdocs/nft/index.jsp?show=s9) in the NFT Reference Manual.

File-Transfer Task	**NFT Command**
GENERAL	
Change remote directory to *newdir* on aaa	cd aaa:*newdir*
Change local directory to *newdir*	cd :*newdir*
Change storage(*) directory to *newdir*	cd *newdir*
List remote directory contents on aaa	dir aaa:
List local directory contents	dir :
List storage(*) directory contents	dir
Put (copy) local file *t1* to remote host aaa as *t2*	cp :*t1* aaa:*t2*
Get (copy) remote file *t3* from aaa as local file *t4*	cp aaa:*t3* :*t4*
Transfer file *t5* on aaa to file *t6* on bbb (both remote)	cp aaa:*t5* bbb:*t6*
Delete remote file *t3* from storage	delete *t3*
STORAGE DEFAULTED(*)	
Store local file *t1*	put *t1*
Store local file *t1* as *t2*	put *t1 t2*
Retrieve from storage file *t3*	get *t3*
Retrieve *t3* as local file *t4*	get *t3 t4*
Add ACL(+) to stored file *t1*	acladd *acl t1*
Restore default ACL(+) to stored file *t1*	aclclear *t1*
Remove ACL(+) entry from stored file *t1*	aclremove *acl t1*
Replace ACL(+) entry for stored file *t1*	aclreplace *acl t1*
Display ACL(+) for stored file *t1*	aclshow *t1*
CONTROL OPTIONS	
Prevent all overwriting (default)	noclobber
Allow overwriting (for updates)	clobber
Start a log of NFT actions	log *logfile*
Close the log file	clog
Change default remote host(*)	open *host*
Terminate NFT	quit (*)

(*)You can change NFT's default remote host from storage to something else by using the OPEN command, but you should consult the OPEN command section (URL: http://www.llnl.gov/LCdocs/nft/index.jsp?show=s9.26b) of the NFT manual before you rely on it.
(+)Access control list, for more fine-grained control than with UNIX permissions.

NFT Example

This annotated example shows typical file transfers using NFT.

GOAL: To transfer files from a secure LC machine to STORAGE without logging on to all of the machines, using NFT.

STRATEGY: (1) Start NFT. Notice that unlike FTP, you do not log on to any particular remote host to "open a connection."
(2) Use storage-defaulted command PUT to transfer file t1 from the client machine (where NFT runs) to storage.llnl.gov as file t2. Note that NO hosts are specified in this command because the default location is STORAGE.
(3) Try to retrieve file t2 from STORAGE to local file t1 using the storage-defaulted GET command. Becaue NFT's default environment is NOCLOBBER, this attempt fails (t1 already exists). You could use the CLOBBER option next, to allow this overwrite, or...
(4) Use GET to retrieve t2 from storage with no name change (and hence no overwriting of t1).

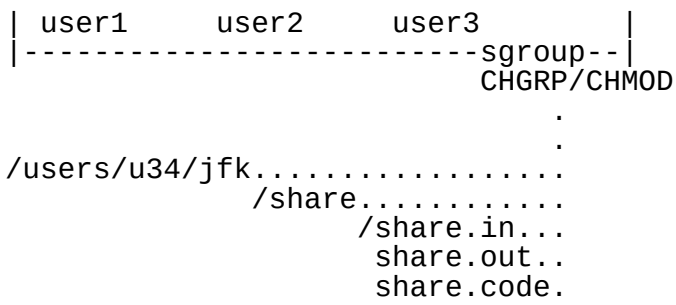
```
(1)  nft
(2)  nft>put t1 t2
     4.0. 95 bytes sent in 1.0 seconds
     (0.1 Kbytes/s) from /g/g0/jfk/t1 to ~/t2
(3)  nft>get t2 t1
     5.0. error. Cannot clobber existing sink.
     /g/g0/jfk/t1
(4)  nft>get t2
     6.0. 95 bytes received in 1.8 seconds
     (0.1 Kbytes/s) from ~/t2 to /g/g0/jfk/t2
     nft>quit
```

Sharing Stored Files

Sharing some stored files with one or several other users is one of the most common storage goals yet one of the hardest to achieve. This section explains the logic and reveals the many tedious steps needed for sharing stored files. (You may want to consider using other file-sharing techniques available on LC production machines. Consult the "File-Sharing Alternatives Compared" section of EZFILES (URL: http://www.llnl.gov/LCdocs/ezfiles) for an overt analysis of several choices.)

All sharing of stored files on LC's HPSS system happens by means of storage groups. You and those with whom you want to share stored files must first find or create a storage group to which you all belong, you must assign the files to be shared and every parent directory of them to that common storage group, and you must open the file and directory permissions (of the whole tree) to allow group reads (executes, or writes). Even then, file-sharing is brittle and error prone.

The (sample) users, groups, directories, and files diagrammed here will be used to illustrate the file-sharing steps described in the subsections below:



Using Storage Groups

A group is just a named set of users that agree among themselves to optionally allow (some of their) files to be readable, or even writable, by all group members. Unfortunately, at LC online groups (e.g., on GPS17) and storage groups are managed independently, and a file loses its group status at the time you store it. So you must arrange the sharing of stored files by working exclusively with storage groups, regardless of what separate online group arrangements you may have.

To discover the storage group(s) to which you belong, use the DCECP utility (Distributed Computing Environment Control Program) on any LC production machine, as shown here (where *uname* is your user name):

```
User: dcecp
R/Us: dcecp> user show uname
Rtne: {fullname {Leonora Florestan}}
      [four other data lines...]
      {groups group1, group2, group3...}
      [many other data lines...]
R/Us: dcecp> quit
```

In this report, *group1* is always the single-member group with the same name as your login name (and on open HPSS it is often your only storage group).

To discover which other users also belong to any specific storage group (such as *gname*), use DCECP again with a different command line:

```
User: dcecp
R/Us: dcecp> group list gname
Rtne: /.../server/user1
      /.../server/user2
      /.../server/user3
      [other data lines...]
R/Us: dcecp> quit
```

If you and the others with whom you intend to share stored files all belong to at least one storage group, as revealed by running DCECP, then you can use that group and the commands in the next subsection to enable file sharing in HPSS. If, however, the set of users who want to share stored files has no existing storage group in common, then you must create one with the appropriate membership by using LC Form SCF-2 (Create/Update Group) before you can take any further steps. Contact the LC Hotline; there is no online or automatic way to avoid this group-creation paperwork.

Setting Stored-File Permissions by Group

Once you have the files you want to share and the name of a storage group to whom all sharing users belong (see the previous subsection), you can follow these steps, all involving (somewhat unusual) FTP commands, to enable the sharing of stored files:

(1) Open an FTP session to STORAGE.
All file-sharing arrangements require passing group and permission information to the storage system using the indirect mechanism that FTP provides for such nonstandard activity.

```
ftp storage
```

(2) Create a storage directory to hold the shared files.
In this example, the shared-files directory is called *share* and the shared file is called *share.code* (see also the figure at the start of the file-sharing section), but these can obviously be generalized as you need. In your FTP session type

```
mkdir share
```

(3) Assign your storage home directory to the share group.

If your default arrival directory in storage is /users/u34/jfk and if the storage group containing all the file-sharing users is *sgroup*, then use this indirect FTP command

```
quote site chgrp sgroup /users/u34/jfk
```

to associate the two. One side effect is that you cannot share with two different groups at once. (You can also change storage groups for any of these steps by using the special CHGRPSTG (page 35) tool, described in a later section.)

(4) Assign your file-sharing directory to the share group.

Since you made the share directory as a child of /users/u34/jfk in step (2), you can now associate it too with the file-sharing storage group *sgroup*:

```
quote site chgrp sgroup share
```

(5) Assign group permisssions to the file-sharing directory.

To allow other members of storage group *sgroup* to read, write, and execute (list) the file(s) in the share directory, use this indirect FTP command

```
quote site chmod 775 share
```

to expand its default group permissions. (You can also change storage permissions for any of these steps by using the special CHMODSTG (page 32) tool, described in a later section.)

(6) Store the files to be shared.

If you move (CD) to the file-sharing directory and PUT the file(s) to be shared, they will lose their online permissions but they will arrive associated with the share group *sgroup*, which they inherit from the file-sharing directory:

```
cd share
put share.code
[more puts if there are more files to share]
```

(7) Assign group permissions to the file(s) to be shared.

Even if their online permissions allowed sharing by group, storing the file(s) erased those decisions. So as with step (5) above, you need to declare the availablity of each file to the members of *sgroup*:

```
quote site chmod 775 share.code
```

Because of the complexity of this process, some LLNL project teams have successfully petitioned the LC Storage Group to create ad hoc, top-level directories through which they share stored files.

Reading Shared Stored Files

After you have used the previous two subsections to enable others in storage group *sgroup* to share the file(s) in the share directory, they can follow these steps to retrieve those file(s):

```
ftp storage
cd /users/u34/jfk/share
get share.code
```

Note that impatient attempts to directly GET file /users/u34/jfk/share.code (while in another storage directory) may misleadingly fail with the message "no such file or directory." Even when successful, sharing stored files in LC's HPSS archive is not trivial.

Macintosh File Transfer Problems

You can transfer files to or from an Apple Macintosh computer using FTP. The easiest, most reliable way is to run an FTP client (such as Fetch) on the Macintosh itself. However, if the Macintosh lies outside the llnl.gov domain, you will need to install and run a Virtual Private Network (VPN) client on your machine to enable your FTP traffic to pass through LLNL's protective firewall. Alternatively, you can log on to an LC machine (within llnl.gov) and execute an FTP client there that connects back to your Macintosh. To do this your Macintosh computer will need an FTP server. You may need the help of your system administrator to buy, install, and configure an FTP server for your Macintosh. (NCSA Telnet includes an FTP server and is configured using the EDIT | PREFERENCES menu-bar choice.) See the Accessing Storage (page 10) section above for some details; see the Firewall and SSH Guide (URL: http://www.llnl.gov/LCdocs/firewall) for full details.

When transferring files between a Macintosh desktop computer and STORAGE two problems arise:
(1) Macintosh files have an unusual format that sometimes interferes with successful or appropriate file transfer, especially for remote FTP clients.
(2) Some Macintosh file names are unsuitable for use after transfer to a UNIX machine. You may need to change these names before (or during) transfer.

Macintosh File Format

Macintosh files consist of two parts, the data fork and the resource fork. In most (but not all) cases, one fork is empty. Data files tend to use only the data fork, while executables tend to use only the resource fork. This table shows the situation:

	Data Fork (text/binary data)	**Resource Fork (executables)**
FTP Transfer		
.....ASCII	gets/puts text, with conversion	
.....binary	gets/puts images, no conversion	[needs Macbinary conversion first]

The table also shows how these two-part files interact with FTP. FTP run in ASCII mode copies text data to or from the data fork of a Macintosh file, with automatic format conversion. FTP run in binary mode copies binary data (such as images) to or from the data fork, with no conversion.

Remote FTP clients never copy the contents of the resource fork unless you have previously converted the file to a special format called Macbinary (which forces both data and resource forks of the original file into the data fork of the converted file, to enable transfer). To convert and then transfer a Macintosh file without spoiling the text data and nonexecutable binary data (such as GIF images or word processing content) already in the data fork often requires special preprocessing. The SUNTAR freeware program for the Macintosh, akin to TAR on UNIX machines, is one reliable way to meet this need.

Suntar's Role

SUNTAR is a Macintosh freeware program modestly named after its creator ("Speranza's un-tar"). SUNTAR bundles and unbundles "archives" of files on a Macintosh, optionally converting their format in the process, much like TAR on a UNIX system. Using SUNTAR before you transfer files from a Macintosh to STORAGE (or after you return them to a Macintosh from STORAGE) lets you:
(1) Control the format conversions that may otherwise unexpectedly occur during FTP file transfers.
(2) Control what parts of Macintosh files are transferred, so nothing is lost.
(3) Enable the successful transfer of Macintosh executables and other special-format files (such as word processing files) to LC storage, and their undamaged return later.

SUNTAR is available from several anonymous FTP servers around the world, including:

```
   mirror.aol.com
username: ftp [NOT anonymous]
password: your e-mail address
mode: binary
CD to: /pub/info-mac/cmp
get: suntar-221.hqx
```

Executing SUNTAR offers a default configuration that you can accept for easy use. Brief documentation comes with the program in two SimpleText files.

To transfer files from a Macintosh using SUNTAR and FTP to avoid the two-fork problem explained above, follow these steps:
(1) On the Macintosh, execute SUNTAR.
(2) Select FILE | NEW ARCHIVE from SUNTAR's menu bar to create a new tar (archive) file, whose name you can specify.
(3) Select WRITE | WRITE TAR FILE from the menu bar, and, when prompted for your choice, pick the same file that you created in step (2).
(4) Select WRITE | WRITE *format* and each file that you pick will be converted to the *format* that you specify (if necessary) and then copied into the archive you selected in step (3). Use Macbinary format for files that will not be opened on any other machine; avoid Macbinary if you really need to extract the file using TAR once you have moved the archive to a UNIX system.
(5) Quit SUNTAR.
(6) On a remote machine, execute FTP. Then connect to your Macintosh, request BINARY mode, and GET the archive file that you created and stocked using SUNTAR. You can store this file (use BINARY again) or open it with TAR, but any Macbinary- encoded files within it cannot be used except on a Macintosh.

To return a SUNTARed archive file from STORAGE to a Macintosh (or another LC UNIX machine) and restore its component files for use, follow these steps:
(1) On a remote machine, execute FTP. Then connect to your Macintosh, request BINARY mode, and PUT the archive file that you (previously) created and stocked using SUNTAR. Quit FTP.
(2) On the Macintosh, execute SUNTAR.
(3) Select FILE | OPEN FILE/DECODE from the menu bar and then pick the archive file that you just FTPed.
(4) Select FILE | LIST to report the contents of the archive file in a "console window," quite the way TAR lists contents under UNIX.

(5) Select FILE | EXTRACT SELECTED FILES to pick from a menu of files within the archive the one(s) to extract (and convert if necessary), into a folder that you specify. These are now ready for use on the Macintosh, as if they had not been transferred.
(6) Select FILE | QUIT to end SUNTAR when your file extractions are finished.

Macintosh File Name Problems

The Macintosh OS supports file names that contain blanks (spaces) or nonASCII characters (for example,

```
Memo to Fred
```

is an acceptable Macintosh file name). Names containing such characters are not supported by UNIX and will cause file loss or serious mishandling on LC's UNIX production machines, on the UNIX- based storage system, and on the open-secure File Interchange Service with UNIX nodes on each end. You must therefore change all blank-containing Macintosh file names to consist of ASCII characters without blanks, such as

```
yMemoToFred
memo.to.fred
memo_to_fred
```

before (or during) the transfer of these files from a Macintosh to a UNIX machine.

If you are able to run Fetch, the FTP Macintosh client, you can change a file's name during the transfer process itself:
(1) Execute Fetch on your Macintosh.
(2) Click the PUT FILE button, highlight the (Macintosh) name of the file you want to transfer from the offered list, and click the OPEN button.
(3) When the dialog box appears offering to "save file on *unix*.gov as:" then
(a) delete the Macintosh name containing blanks,
(b) supply (type in) a suitable UNIX name without blanks, and
(c) click on OK to transfer the file so it arrives with its new name on the UNIX target machine.

If instead you run an FTP client on a remote (UNIX) machine to transfer files from your Macintosh, you must change all problematic file names BEFORE you begin the transfer process (quoting a file name containing blanks is NOT adequate compensation for most FTP clients). On the Macintosh, open each file with a word processor and use the FILE | SAVE AS menu choice to make a copy whose new name contains no blanks or nonASCII characters. Then execute FTP on the remote machine and transfer from the Macintosh the newly renamed copies for reliable use under UNIX.

Storage Assistance Tools

LC's Tru64 Compaq computers (for example, the nodes of the open GPS cluster and the secure SC cluster), all the massively parallel IBM production machines, and also the Linux/CHAOS machines offer three public, user-developed programs to handle three common storage tasks more conveniently than is possible with FTP or NFT. In fact, these storage assistance tools perform some helpful tasks (such as recursive changes on stored files) not possible with FTP (NFT offers a suboption, -R, that you can invoke to recursively change stored files).

These special storage tools and their roles are:

lstorage	lists your storage directories and stored files in any of several formats, recursively if you request.
chmodstg	changes the UNIX permissions on your storage directories or your stored files, recursively and symbolically if you request.
chgrpstg	changes the (storage) group for your storage directories or your stored files (to enable file sharing), recursively if you request.

All are located in /usr/local/bin on the machines where they they have been installed (so most users can run them just by typing their names).

WARNING: Because all three storage-assistance tools are really Perl scripts, they yield very verbose and confusing error messages if you happen to run them when the LC storage system (either open or secure) is offline for maintenance.

In addition, LC provides a special-purpose front-end to parallel FTP that is customized to very efficiently store and retrieve large archive (TAR-format library) files. This combination file bundler and fast STORAGE interface is called HTAR. A short subsection below introduces HTAR's features and syntax, while LC's separate HTAR Reference Manual (URL: http://www.llnl.gov/LCdocs/htar) gives a thorough analysis of both good usage and known pitfalls. HTAR also offers the unique ability to retrieve a member file from within a still-stored archive, even without staging the archive from tape to disk in HPSS.

LSTORAGE (List Stored Files)

EXECUTE LINE.

LSTORAGE lists your storage directories and the files that they contain. To run LSTORAGE on the LC production machines where it is installed, type

`lstorage` [*`options`*] [*`dirnames`*]

By choice of LSTORAGE options you can specify output format (single or multiple colunms), output scope (local or recursive), and level of detail (names only or other information too). The basic pattern for using LSTORAGE options is:

```
              Recursive   Nonrecursive
             -------------------------
            |
Single      | -lR, -j       -l
column      |
            |
Multiple    |  -R           default,
column      |               -C
```

Because LSTORAGE runs noninteractively, redirecting its output to a file for later reuse is easy (e.g., lstorage > *outfile*).

DEFAULTS.

Without a specified directory, LSTORAGE reports on your top-level ("home") storage directory. Without options, LSTORAGE lists (only) the names of files and directories contained in the specified storage directory, in multiple columns. If you specify a space-delimited list of several target storage directories (all names relative to your home storage directory), LSTORAGE reports on each one in the order in which you listed them on the execute line.

SPECIAL BENEFITS.

LSTORAGE takes the place of using FTP's DIR or LS options. Unlike FTP, LSTORAGE avoids the long warning message, can make recursive reports, is easy to redirect, and can report on several storage directories at once.

TYPICAL USES.

lstorage -lR > storage.list

> places into the file storage.list a detailed, recursive report on all of your storage directories and stored files (and their properties), starting with your "home" storage directory and working down the tree.

lstorage -j project2/admin

> lists the names (only) of your storage directories, subdirectories, and stored files starting with the project2/admin directory and continuing recursively downward through the tree. The list is a single column indented at every new level to reveal nesting.

OPTIONS.

Scope options:

-a	lists all directories and files, including those whose names begin with a dot(.). NOTE: on the one hand, listing stored files such as .cshrc is default behavior for LSTORAGE even without invoking -a; on the other hand, even with -a invoked the list still omits the single and double dot (. and ..) entries that FTP's DIR reports.
-l	(lowercase ell) lists in long format, with details on the permissions and groups for every storage directory or stored file covered in the report.
-R	recursively includes all the children (subdirectories and stored files) of the directory specified on the execute line (compare with -j).

Format options:

-C	(default) lists storage directories and stored files in multiple columns with entries sorted down the columns.
-j	lists storage directories and stored files recursively (entails -R) in a single column with nesting revealed by extra indenting (names only).
-h	displays the LSTORAGE help package (a brief list of options). Help cannot be combined with any other options.
-t *sss*	sets the LSTORAGE timeout to *sss* seconds (default timeout is 300 seconds).

CHMODSTG (Change Storage Permissions)

EXECUTE LINE.

CHMODSTG changes the permissions on your storage directories or your stored files. To run CHMODSTG on the LC production machines where it is installed, type

chmodstg [*options*] [*dirname*]

By choice of CHMODSTG options you can specify the desired permissions for a specific storage directory, a specific stored file, all files in a directory, or (recursively) all children of a specific directory to all levels. You can also specify uninterrupted, noninteractive changes or instead request interactive prompting for your desired permissions and files (with optional report on each change made). The basic pattern for using CHMODSTG options is (all except -s can be combined with -R for recursive scope):

	Prompts		No prompt
	For perms only	For perms and files	
No reports	default, -f, -d, -s	-i	-F *perm* -D *perm*
Report results	-v or lstorage -l		-v or lstorage -l

DEFAULTS.

Without a specified directory, CHMODSTG acts on your top-level ("home") storage directory. Without permission-related options (e.g., chmodstg -R dir1), CHMODSTG prompts for your desired directory and file permissions and then changes both with no confirmation.

SPECIAL BENEFITS.

CHMODSTG takes the place of using FTP's QUOTE SITE CHMOD indirect command. Unlike FTP, CHMODSTG avoids the long warning message, can make recursive changes, and accepts symbolic rather than only octal permissions.

TYPICAL INTERACTIVE USES.

chmodstg -R project2/admin

announces that CHMODSTG will act recursively starting from the specified directory, prompts for your desired permissions on storage directories, prompts (separately) for your desired permissions on stored files, then changes the permissions without confirmation.

chmodstg -iR project2/admin

same as above (for -R), but also prompts for your yes/no choice for each directory and file processed.

chmodstg -ivR project2/admin

same as above (for -iR), but also reports the specific change made for every file (e.g., "changed from 750 to 700") as it occurs.

TYPICAL NONINTERACTIVE USES.

chmodstg -D775 project2/admin

assigns permission 775 to all subdirectories of the specified storage directory (use -s to change that directory itself), without prompting or confirmation.

chmodstg -D750 -F650 -R project2/admin

starts at the specified directory and recursively assigns 750 to every subdirectory and 650 to every stored file encountered as it works down the tree, without prompting or confirmation.

SPECIFYING PERMISSIONS:

CHMODSTG accepts permissions as either three-digit octal numbers (exactly three digits, no spaces) or as a comma-delimited list of symbolic triples (e.g., u+x,g-w) built up from the UNIX components [augo], [+-], and [rwx]. (FTP accepts only the octal format.) Users unfamilar with either style of specifying permissions can read a concise, overtly diagrammed summary of both in the "How to Specify Permissions" section of the EZFILES (URL: http://www.llnl.gov/LCdocs/ezfiles) basic guide.

OPTIONS.

Permission options:

-F*perm* specifies (in either octal or symbolic format) the UNIX permissions *perm* to assign to every stored file (but not directories) that CHMODSTG treats during this run, as selected by other options. This disarms the file-permissions prompt.

-D*perm* specifies (in either octal or symbolic format) the UNIX permissions *perm* to assign to every storage directory (but not stored files) that CHMODSTG treats during this run, as selected by other options. This disarms the directory-permissions prompt.

Scope options:

-f changes file permissions only (omits directories). CHMODSTG prompts you for the desired permissions. The default without -f or -d is to change both.

-d changes directory permissions only (omits files). CHMODSTG prompts you for the desired permissions. The default without -f or -d is to change both.

-s *pathname* changes permissions only for the one directory or file specified by its pathname (relative to your home storage directory). Using -s disables all other CHMODSTG options except -v, so CHMODSTG always prompts for your desired permissions even if you include -F or -D on the execute line.

-R recursively includes all the children (subdirectories and stored files) of the directory specified on the execute line. You can combine -R with other options (except -s) to further control CHMODSTG's scope of action.

Interaction options:

-i prompts for your yes/no confirmation for every directory or stored file that CHMODSTG tries to change (regardless of whether you also want prompting for desired permissions). Any response except YES is treated as NO; you can NOT supply different permissions for different files by using -i.

-v interactively reports the permission change made for every directory or stored file that CHMODSTG changes (e.g., "changed from 650 to 700"). You can combine -v with CHMODSTG's various prompting options, or use it for confirmations even without prompts.

-s is a scope option (see above) but always behaves interactively, even if you try to disable its prompts.

-h displays the CHMODSTG help package (a brief list of options). Help cannot be combined with any other options.

CHGRPSTG (Change Storage Groups)

EXECUTE LINE.

CHGRPSTG changes the (storage) group for your storage directories or your stored files. To run CHGRPSTG on the LC production machines where it is installed, type

chgrpstg [*options*] *groupname* [*dirname*]

There is no prompt or default for the desired *groupname*, which you must specify on every CHGRPSTG execute line. (To discover your current storage groups, run DCECP on any LC production machine and reply to its prompt with the request:
user show *yourlogin*
See the Sharing Stored Files (page 21) section for details and examples.) Most CHGRPSTG invocations run noninteractively, but you can request prompting or confirmatory reports, alone or together with recursive execution, by following this pattern of options:

	Recursive	Nonrecursive
Prompts and reports	-ivR	-iv
No prompts or reports	-R	default

DEFAULTS.

Without a specified directory, CHGRPSTG acts on your top-level ("home") storage directory. Without options, CHGRPSTG changes the group for one "layer" in your storage hierarchy (for every member of a specified directory but not the directory itself nor the children of its subdirectories). See the comparative example below.

SPECIAL BENEFITS.

CHGRPSTG takes the place of using FTP's QUOTE SITE CHGRP indirect command. Unlike FTP, CHGRPSTG avoids the long warning message, can make recursive changes, and can (optionally) treat just files or just directories at any level in your storage hierarchy.

TYPICAL INTERACTIVE USES.

chgrpstg -ivR newgrp project2/admin

announces that CHGRPSTG will act recursively starting from the specified directory, prompts for your yes/no choice for each directory and file processed, and reports the specific change made for every file (e.g., "changed from oldgrp to newgrp") as it occurs.

TYPICAL NONINTERACTIVE USES.

chgrpstg -s newgrp project2/admin

(change exactly one directory) changes the storage group only for the single directory specified. Note the different syntax from CHMODSTG (group name precedes pathname).

chgrpstg newgrp project2/admin

(change one "layer," the default) changes the storage group for all files and directories within the specified directory, but not for that directory itself nor for any children of its subdirectories.

chgrpstg -R newgrp project2/admin

(change all layers) changes the storage group for all files and directories within the specified directory, and also for all of its children working recursively down your storage hierarchy.

OPTIONS.

Scope options:

-f changes file groups only (omits directories). The default without -f or -d is to change both.

-d changes directory groups only (omits files). The default without -f or -d is to change both.

-s *groupname pathname*

changes groups only for the one directory or file specified by its pathname (relative to your home storage directory). Using -s disables all other CHGRPSTG options except -v. Note the syntax difference from CHMODSTG: here, the group name precedes the pathname immediately after -s.

-R recursively includes all the children (subdirectories and stored files) of the directory specified on the execute line. You can combine -R with other options (except -s) to further control CHGRPSTG's scope of action.

Interaction options:

-i prompts for your yes/no confirmation for every directory or stored file that CHGRPSTG tries to change. Any response except YES is treated as NO; you can NOT supply different groups for different files by using -i.

-v interactively reports the group change made for every directory or stored file that CHGRPSTG changes (e.g., "changed from oldgrp to newgrp"). You can combine -v with CHGRPSTG's -i prompting option, or use it for confirmation reports even without prompts.

-h displays the CHGRPSTG help package (a brief list of options). Help cannot be combined with any other options.

HTAR (Manage Stored File Collections)

ROLE.

On LC production machines (but not at other ASCI sites), HTAR is a separate, locally developed utility program that serves as a special-purpose front end to PFTP for storage access. HTAR combines a flexible file bundling tool (like TAR) with fast parallel access (PFTP) to open and secure STORAGE, to let you store and selectively retrieve even very large sets of files very efficiently. (Invoking HTAR's -F option lets you generalize these features for fast, file-bundled transfer to *non*STORAGE locations as well.)

FEATURES.

HTAR's enhanced features include:

- Uses a TAR-like syntax and supports TAR-compatible archive files by relying on the POSIX 1003.1 TAR file format.
- Bundles files in memory using multiple concurrent threads and transfers them into an archive file built *directly* in storage by default, to avoid needing extra local online disk space.
- Takes advantage of available parallel interfaces to storage to provide fast file transfers (measured at as high as 150 Mbyte/s, over 30 times the typical rate for transferring small files separately).
- Uses an external index file to easily accommodate thousands of small files in any archive, and to support retrieval of specified files from within a still-stored archive without first retrieving the whole archive from HPSS. (WARNING: you can use filters such as * to *create* an HTAR archive but you CANNOT reliably use filters to *retrieve* files from within an already stored HTAR archive. See the "Retrieving Files" section of the HTAR Reference Manual (URL: http://www.llnl.gov/LCdocs/htar) for possible workarounds.)
- Imposes no limit on the total size of the archives that it builds (some have successfully reached 200 Gbyte) and accepts input files (archive members) as large as 8 Gbyte.

EXECUTE LINES.

When the storage system (HPSS) is up and available to users you can execute HTAR with a command line that has the general form

```
htar action archive [options] [filelist]
```

and the specific form

```
htar -c|t|x|X -f archive [-BdEFhHILmMopSTvVwY] [flist]
```

where exactly one *action* and the *archive* are always required, while the control options and (except when using -c) the *filelist* can be omitted (and the options can share a hyphen flag with the action for convenience). Users familiar with TAR can guess how to run HTAR from this model (although there are some tricky syntax differences). Others should consult the HTAR Reference Manual (URL: http://www.llnl.gov/LCdocs/htar) for usage suggestions, annotated examples, technical tips, full option details, and known problems.

One unusual feature of HTAR lets you not only avoid retrieving an entire archive from storage before extracting specified member files from within it, but also lets you (optionally) avoid even staging tape-resident archive files to HPSS disk before extracting specified members directly to your local machine. The NOSTAGE suboption of HTAR's -H (uppercase) control option lets you quickly retrieve (small) files

from within a (much larger) stored-on-tape archive file, while leaving the archive on tape. For example, to retrieve file TEST5 from within the archive MYPROJ.TAR, stored in the PROJECTS subdirectory of your HPSS home directory, while leaving the whole archive still stored on tape, you could use

```
htar -x -f projects/myproj.tar -H nostage test5
```

Disclaimer

Keyword Index

To see an alphabetical list of keywords for this document, consult the next section (page 42).

Keyword	Description
entire	This entire document.
title	The name of this document.
scope	Topics covered in EZSTORAGE.
availability	Where these programs run.
who	Who to contact for assistance.
introduction	Role and goals of EZSTORAGE.
overview	LC storage strengths and weaknesses.
storage-summary	Storage system constraints and common commands.
storage-interfaces	Descriptions of STORAGE access pathways.
ftp-overview	Brief description of FTP interface.
nft-overview	Brief description of NFT interface.
additional-interfaces	Alternative interface options.
accessing-storage	Offsite access strategies compared.
storage-copies	Multiple copies of same stored file.
ftp	Using FTP to transfer files.
file-transfer-protocol	Using FTP to transfer files.
ftp-commands	Basic FTP options explained.
ftp-example	Sample file transfer with FTP.
ftp-pitfalls	Wildcard dangers with stored files.
nft	Using NFT to transfer files.
network-file-transfer	Using NFT to transfer files.
nft-syntax	Specifying sec. levs, hosts with NFT.
nft-commands	Basic NFT options by task.
nft-example	Sample file transfer with NFT.
sharing-files	Sharing stored files.
file-sharing	Sharing stored files.
storage-groups	Using storage groups.
permissions	Setting permissions by storage group.
reading-shared-files	Sharing by the reader.
macintosh-problems	Known Macintosh file-transfer problems.
file-format	Two fork Macintosh file format.
suntar	Macintosh suntar aids file transfers.
file-name-problems	Changing file names containing blanks.
storage-tools	Three LC storage-helper tools.
lstorage	Tool to list stored files.
chmodstg	Tool to change storage permissions.
chgrpstg	Tool to change storage groups.
htar	Tool to bundle files into storage archives.
index	The structural index of keywords.
a	The alphabetical index of keywords.
date	The latest changes to this document.
revisions	The complete revision history.

Alphabetical List of Keywords

Keyword	Description
a	The alphabetical index of keywords.
accessing-storage	Offsite access strategies compared.
additional-interfaces	Alternative interface options.
availability	Where these programs run.
chgrpstg	Tool to change storage groups.
chmodstg	Tool to change storage permissions.
date	The latest changes to EZSTORAGE.
entire	This entire document.
file-format	Two fork Macintosh file format.
file-name-problems	Changing file names containing blanks.
file-sharing	Sharing stored files.
file-transfer-protocol	Using FTP to transfer files.
ftp	Using FTP to transfer files.
ftp-commands	Basic FTP options explained.
ftp-example	Sample file transfer with FTP.
ftp-overview	Brief description of FTP interface.
ftp-pitfalls	Wildcard dangers with stored files.
htar	Tool to bundle files into storage archives.
index	The structural index of keywords.
introduction	Role and goals of EZSTORAGE.
lstorage	Tool to list stored files.
macintosh-problems	Known Macintosh file-transfer problems.
network-file-transfer	Using NFT to transfer files.
nft	Using NFT to transfer files.
nft-commands	Basic NFT options by task.
nft-example	Sample file transfer with NFT.
nft-overview	Brief description of NFT interface.
permissions	Setting permissions by storage group.
reading-shared-files	Sharing by the reader.
nft-syntax	Specifying sec. levs, hosts with NFT.
overview	LC storage strengths and weaknesses.
revisions	The complete revision history.
scope	Topics covered in EZSTORAGE.
sharing-files	Sharing stored files.
storage-copies	Multiple copies of same stored file.
storage-groups	Using storage groups.
storage-interfaces	Descriptions of STORAGE access pathways.
storage-summary	Storage system constraints and common commands.
storage-tools	Three LC storage-helper tools.
suntar	Macintosh suntar aids file transfers.
title	The name of this document.
who	Who to contact for assistance.

Date and Revisions

Revision Date	Keyword Affected	Description of Change
17Feb04	storage-summary	COS and ACL commands added.
	ftp-pitfalls	Recursive NFT deletes now enabled.
	nft-syntax	Five new ACL commands noted.
	nft-commands	Five ACL commands added to table.
	nft-example	Dialog updated, details added.
18Nov03	htar	Speed and size details updated.
	storage-summary	Maximum file size updated.
11Aug03	storage-summary	Cross ref. to HTAR added.
	ftp-commands	Cross ref. to HTAR added.
	nft-syntax	Cross ref. to HTAR added.
	htar	NOSTAGE feature explained, illustrated.
12May03	introduction	NETMON units user controlled now.
	ftp-commands	PARALLEL has altered role.
	storage-tools	Now under Linux/CHAOS too.
19Feb03	introduction	SFTP is not a storage interface.
	overview	SFTP is not a storage interface.
	additional-interfaces	SFTP is not a storage interface.
21Nov02	introduction	FTP monitored by class of service now.
	ftp-commands	FTP monitored by class of service now.
	storage-copies	Another role for COS noted.
08Oct02	storage-tools	Forest departs, availability clarified.
17Jun02	ftp-pitfalls	New section with MDELETE warning.
	ftp-commands	DELETE, MDELETE added.
	nft-commands	DELETE added.
	storage-copies	New section on duplicate copies.
	storage-summary	Maximum file size FTP vs. HTAR.
	index	New keywords for new sections.
02May02	htar	Warning added on HTAR retrievals.
05Feb02	introduction	NETMON cross reference added.
	ftp-commands	NETMON cross reference added.
	ftp-example	Example dialog updated.
	storage-tools	Not available under Linux.
	storage-groups	WEST reference replaced.
27Aug01	overview	HTAR role added.
	scope	HTAR availability noted.
	storage-tools	HTAR role added.
	htar	New summary section added.

	index	New keyword for new section.
09Jul01	overview	PFTP role added.
	storage-interfaces	Automatic parallel FTP to storage. PFTP for storing files elsewhere.
	ftp-commands	PARALLEL local command added.
	ftp-example	Automatic parallel transfers shown.
17Apr01	overview	VPN role noted for offsite FTP.
	accessing-storage	Three strategies revised re VPN, IPA.
	macintosh-problems	VPN role noted for offsite FTP.
26Mar01	ftp-overview	Parallel client now the default.
	ftp-commands	Parallel client now the default.
	ftp-example	Verbose, preauthenticated parallel dialog.
	nft-syntax	No verbosity change for NFT.
25Sep00	sharing-files	Link to file-sharing comparison added.
01Aug00	storage-tools	Now on all LC production machines.
22May00	storage-tools	LSTORAGE, CHMODSTG, CHGRPSTG explained.
	introduction	New sections cross referenced.
	sharing-files	Relevant new tools cited.
	index	New keywords for new sections.
11Apr00	availability	Revised print-file instructions.
	overview	FTP gateway discontinued.
	accessing-storage	FTP gateway discontinued.
10Jan00	entire	Revised first edition of LC EZSTORAGE.
13Dec99	entire	Draft edition of LC EZSTORAGE manual.

TRG (17Feb04)

UCRL-WEB-200719

TRG (17Feb04) Contact: lc-hotline@llnl.gov